```
# Bagging 演示

from sklearn.datasets import make_classification
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.metrics import classification_report
from sklearn.cross_validation import train_test_split

n_f = 30                    # 特征个数
inf_f = int(0.6 * n_f)      # 60%含信息特征
red_f = int(0.1 * n_f)      # 10%冗余特征(含信息特征的线性组合)
rep_f = int(0.1 * n_f)      # 10%重复特征(随机取自含信息特征和冗余特征)

# 创建分类数据集
X,y = make_classification(
    n_samples=500,          # 实例个数
    flip_y=0.03,            # 随机抽 3%实例改变类别,产生一些噪声
    n_features=n_f,
    n_informative=inf_f,
    n_redundant=red_f,
    n_repeated=rep_f,
    random_state=7)

# 划分为训练集,验证集,测试集
X_train,X_test_all,y_train,y_test_all =
    train_test_split(X,y,test_size=0.3,random_state=9)
X_dev,X_test,y_dev,y_test =
    train_test_split(X_test_all,y_test_all,test_size=0.3,
        random_state=9)

# kNN 模型
knn = KNeighborsClassifier()
knn.fit(X_train,y_train)

# 在训练集上预测
y_pred = knn.predict(X_train)
print classification_report(y_train,y_pred)
             precision    recall  f1-score   support

          0       0.88      0.87      0.88       181
          1       0.87      0.88      0.87       169

avg / total       0.87      0.87      0.87       350
```

```
# bagging 集成
bagging = BaggingClassifier(
    KNeighborsClassifier(),
    n_estimators=100,         # 构造 100 个 kNN 模型
    bootstrap=True,
    max_samples=1.0,          # Bootstrap 样本大小用所有实例
    bootstrap_features=True,
    max_features=0.7,         # Bootstrap 特征使用 70%,各模型不一样
    random_state=9)
bagging.fit(X_train,y_train)

y_pred = bagging.predict(X_train)
print classification_report(y_train,y_pred)
             precision    recall  f1-score   support

          0       0.94      0.97      0.95       181
          1       0.96      0.93      0.95       169

avg / total       0.95      0.95      0.95       350

# 看几个模型各自随机抽取的特征,很不一样,所以各模型有变化.
for i,f_set in enumerate(bagging.estimators_features_[0:5]):
    print "estimator %d" % (i+1), f_set
estimator 1 [20 10  6 17 18 11 17  9 14  3 10 10 23 22 18 17 11 21 20  1 16]
estimator 2 [ 3 27 28 20 20 27 25  0 21  1 12 20 21 29  1  0 28 16  4  9 10]
estimator 3 [ 5 23 19  2 16 21  4 13 27  1 15 24  5 14  1  4 25 22 26 29 15]
estimator 4 [23 10 16  7 22 11  0 14 14 17  8 17 27 12 13 23  8  7 27  0 27]
estimator 5 [ 0 26 13 23  7 27 15 18 11 26 18 26  3 22  6 11 21  6 12 19  7]

# 在验证集上预测
y_pred = knn.predict(X_dev)
print classification_report(y_dev,y_pred)
             precision    recall  f1-score   support

          0       0.83      0.84      0.83        51
          1       0.85      0.83      0.84        54

avg / total       0.84      0.84      0.84       105

y_pred = bagging.predict(X_dev)
print classification_report(y_dev,y_pred)
             precision    recall  f1-score   support

          0       0.85      0.88      0.87        51
```

1	0.88	0.85	0.87	54
avg / total	0.87	0.87	0.87	105